

---

# **Web et données liées**

*Version 2014.12.09*

09 December 2014



<b>1</b>	<b>Web de données</b>	<b>1</b>
1.1	Motivation et historique . . . . .	1
1.2	Linked Open Data . . . . .	5
1.3	Forger des IRIs . . . . .	13
<b>2</b>	<b>RDF</b>	<b>15</b>
2.1	Introduction . . . . .	15
2.2	Syntaxe abstraite et sémantique . . . . .	15
2.3	Syntaxes concrètes . . . . .	20
<b>3</b>	<b>SPARQL</b>	<b>25</b>
3.1	Introduction . . . . .	25
3.2	Description du graphe requête . . . . .	26
3.3	Requête SELECT . . . . .	29
3.4	Autres types de requête . . . . .	31
3.5	Quelques requêtes utiles . . . . .	32
<b>4</b>	<b>Vocabulaires et méta-vocabulaires</b>	<b>33</b>
4.1	Vocabulaires . . . . .	33
4.2	Méta-vocabulaires . . . . .	34
4.3	RDF-Schema . . . . .	35
4.4	OWL . . . . .	38
<b>5</b>	<b>Projet 2014</b>	<b>39</b>
	<b>Index</b>	<b>41</b>



## 1.1 Motivation et historique

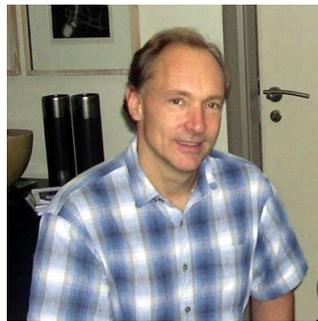


FIGURE 1.1 – (Source image : Wikipedia)  
Tim Berners-Lee

### 1.1.1 Le Web vu par Tim Berners-Lee (1989)

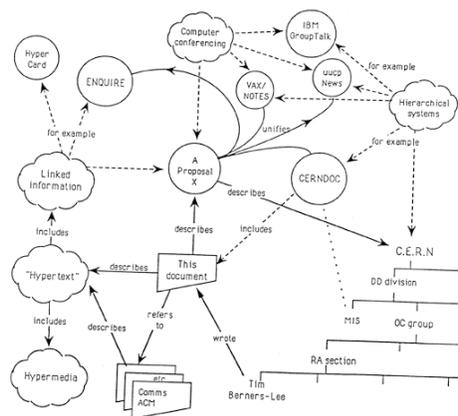


FIGURE 1.2 – Source image : CERN  
« Vague, but exciting »

**Note :** On présente souvent le Web initial comme un web « des documents », par opposition à ses évolutions ultérieures (web social, web sémantique...), mais on voit dans cette figure que lesdites évolutions étaient déjà en puissance dans l'idée de départ.

---

### 1.1.2 Web de ressources

Le web est constitué de **ressources**, par exemple :

- le bulletin météo du jour pour Lyon
- le bulletin météo du jour pour le lieu courant
- ma commande de café de jeudi dernier

Chaque ressource est identifiée par un IRI (*Internationalized Resource Identifier*), e.g. :

- <http://meteo.example.com/lyon>
- <http://meteo.example.com/ici>
- <http://commerce.example.com/commande/192837>

#### /!\ Ressource ≠ fichier

Une ressource n'est pas un simple fichier, dont on récupérerait le contenu. Elle est un objet *actif*, avec lequel on interagit.

- le bulletin météo du jour pour Lyon :
  - son état change chaque jour
- le bulletin météo du jour pour le lieu courant :
  - son état dépend de plus du contexte de l'utilisateur
- ma commande de café de jeudi dernier :
  - on peut agir dessus (par exemple pour l'annuler)

#### Parenthèse : URL/URI/IRI

- URL : Uniform Resource Locator (**RFC 1738**, 1994)
- URI : Uniform Resource Identifier (**RFC 2396**, 1998)
- IRI : Internationalized Resource Identifier (**RFC 3987**, 2005)
- technologies successives
- même concept

#### Ressources et représentations

- Une ressource n'est jamais manipulée directement, mais toujours à travers des **représentations** (pour la créer, la consulter, la modifier).
- Les représentations d'une ressource peuvent varier en fonction
  - de son *état*
  - de l'agent qui manipule la ressource (négociation de contenu, contexte)

représentation :	utilisable par :
texte (HTML...)	humains, moteurs de recherche
médias (image, son...)	<i>surtout</i> humains
<b>données structurées</b>	machines

### 1.1.3 De HTML à XML

XML (eXtensible Markup Language) a été recommandé par le W3C en 1998. L'objectif était de pallier la sémantique « faible » de HTML.

```
<!-- HTML -->
<a href="http://champion.net/">
  Pierre-Antoine <strong>Champion</strong>
  (<em>Maître de conférences</em>) </a>

<!-- XML -->
<Person homepage="http://champion.net/">
  <givenName>Pierre-Antoine</givenName>
  <surname>Champion</surname>
  <job>Maître de conférences</job></Person>
```

### 1.1.4 XML et la sémantique

On a dit tout et son contraire l'apport sémantique de XML :

- XML a *plus* de sémantique que HTML,
- XML a *moins* de sémantique que HTML,

Les deux ont leur part de vérité.

#### XML a *plus* de sémantique que HTML...

... dans le sens où il est extensible : on peut donc exprimer des choses que HTML ne permet pas d'exprimer (e.g. '<givenName>').

- Importance des *espaces de noms*, qui évitent les collisions de noms et fournissent ainsi une sémantique « structuraliste » (i.e. par différenciation).

```
<Person xmlns="http://xmlns.com/foaf/0.1/"
  xmlns:pro="http://example.com/"
  homepage="http://champion.net/">
  <givenName>Pierre-Antoine</givenName>
  <surname>Champion</surname>
  <pro:job>Maître de conférence</pro:job></Person>
```

#### XML a *moins* de sémantique que HTML...

... dans la mesure où :

- un navigateur standard ne saura pas quoi faire de la balise <givenName> ou de la balise <ονομα>,
  - tout au plus il saura les afficher s'il possède une feuille de style,
- tandis qu'il connaît la sémantique de la balise <em> : elle dénote un texte à mettre en évidence *selon les moyens dont il dispose*, par exemple :
  - en le mettant en italique (standard)
  - en le mettant en gras (police déjà en italique)
  - en le mettant en couleurs (police sans italique, terminal)
  - en marquant une pause (synthèse vocale)

## XML : apports et limitations

Le surplus de sémantique promis par XML n'est donc pas « magique » : il suppose

- de créer de nouveaux langages basés sur XML (DTD, schémas),
- d'écrire les logiciels qui *interpréteront* ces nouveaux langages,  
→ chaque langage reste relativement idiosyncratique.

## XML : apports et limitations (suite)

L'apport est donc essentiellement technique : la base commune de XML permet de *factoriser* les efforts de développement et d'apprentissage :

- analyseurs syntaxiques (*parsers*),
- langages de schémas (DTD, XML-Schema, Relax-NG...),
- langages de requêtes (XPath, XQuery),
- langages de transformation (XSL-T),
- méthode de signature cryptographique (xmldsig),
- méthode de compression (EXI)...

### 1.1.5 De XML à RDF

- Le modèle sous-jacent de la syntaxe XML est un arbre (*XML Infoset*), ce qui n'est pas adapté à la structure décentralisée du Web.
- L'objectif du *Resource Description Framework* (RDF), recommandé par le W3C en 1999, vise à munir le Web d'un modèle de données plus adapté, ayant une structure de *graphe*.
- L'objectif est de construire le *Semantic Web* : un web dans lequel les machines ont (enfin) accès à la sémantique des données.
- Recommandation un peu hâtive, présentant quelques défauts importants (notamment l'absence de sémantique formelle).  
→ faible adoption de RDF

### 1.1.6 De RDF à RDF

- En 2004, le W3C publie un nouvel ensemble de recommandations sur RDF pour remplacer celles de 1999.
- Pour des raisons de compatibilité avec l'existant, certains aspects sont conservés malgré les débats qu'ils suscitent, mais les défauts considérés comme majeurs sont corrigés.
- Après cet échec relatif, l'appellation *Semantic Web* tombe peu à peu en disgrâce. Certains défenseurs de RDF parlent plus modestement de *Data Web*, puis de *Web of Linked Data* (2006).
- En 2014, RDF est plus largement accepté. Le W3C publie une version révisée (RDF 1.1), endossant notamment plusieurs syntaxes concrètes.



FIGURE 1.3 – Source image : W3C

### 1.1.7 Le mouvement OpenData

Toute donnée publique (gouvernementale, ONU) ou publiée (scientifique) devrait être accessible sous une forme permettant le traitement automatique (en plus d'une forme lisible pour des humains).





FIGURE 1.5 – Source image : DERI

## 1.2.2 Projet emblématique : DBpedia

- Projet lancé par Chris Bizer en 2007.
- Objectif : extraire les informations structurées (*infobox*) présentes dans Wikipedia pour les exposer en RDF.
- En septembre 2013 (version 3.9) :

The English version of the DBpedia knowledge base currently describes 4.0 million things, out of which 3.22 million are classified in a consistent Ontology, including 832,000 persons, 639,000 places (including 427,000 populated places), 372,000 creative works (including 116,000 music albums, 78,000 films and 18,500 video games), 209,000 organizations (including 49,000 companies and 45,000 educational institutions), 226,000 species and 5,600 diseases.

## Informations structurées dans Wikipedia

The screenshot shows the Wikipedia page for 'Lyon'. At the top, there are navigation tabs for 'article', 'discussion', 'edit this page', and 'history'. Below the title 'Lyon', there is a summary paragraph and a 'Contents' table of contents. The main text describes Lyon as a city in east-central France, mentioning its location, population, and historical significance. On the right side, there is a 'Ville de Lyon' section with the city flag and coat of arms, a motto, and a map of France highlighting Lyon's location. Below the map, there is a table for 'Administration' listing the country, region, department, and other administrative details.

FIGURE 1.6 – Source image : Wikipedia

### 1.2.3 Le « LOD cloud »

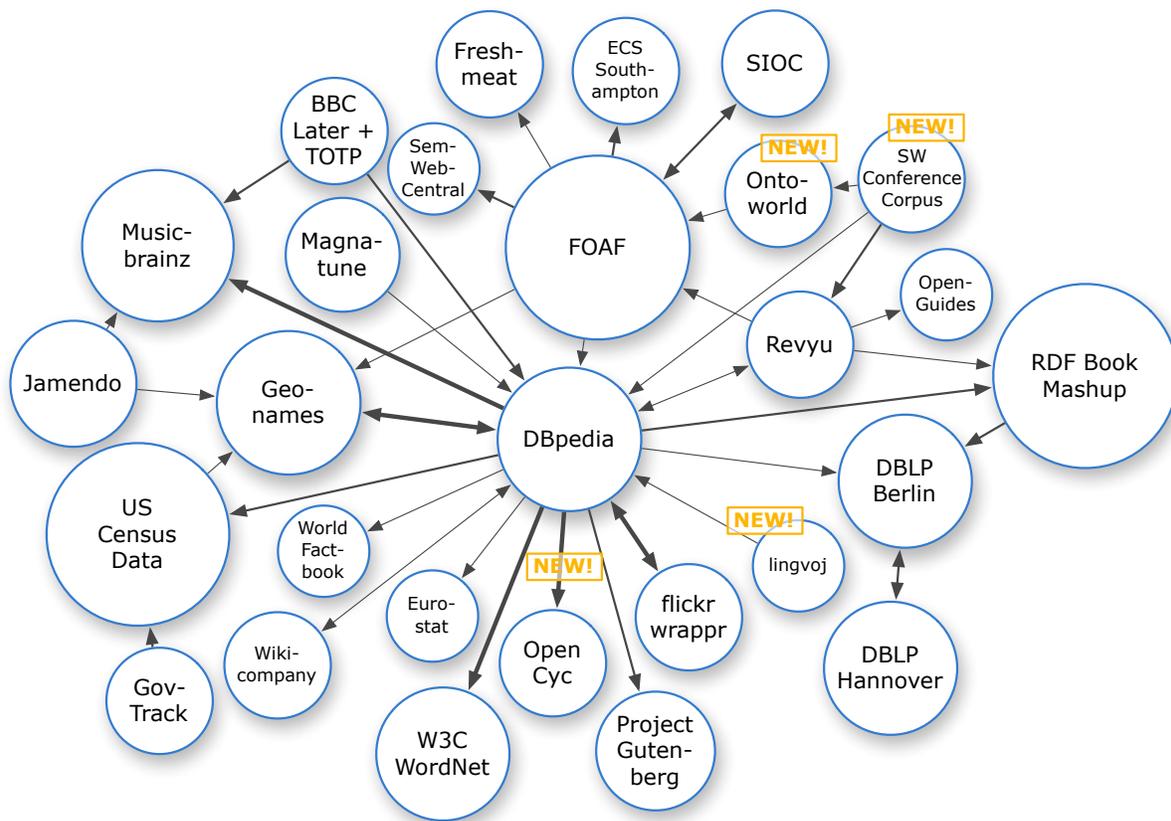


FIGURE 1.7 – Source image : Richard Cyganiak  
En 2007

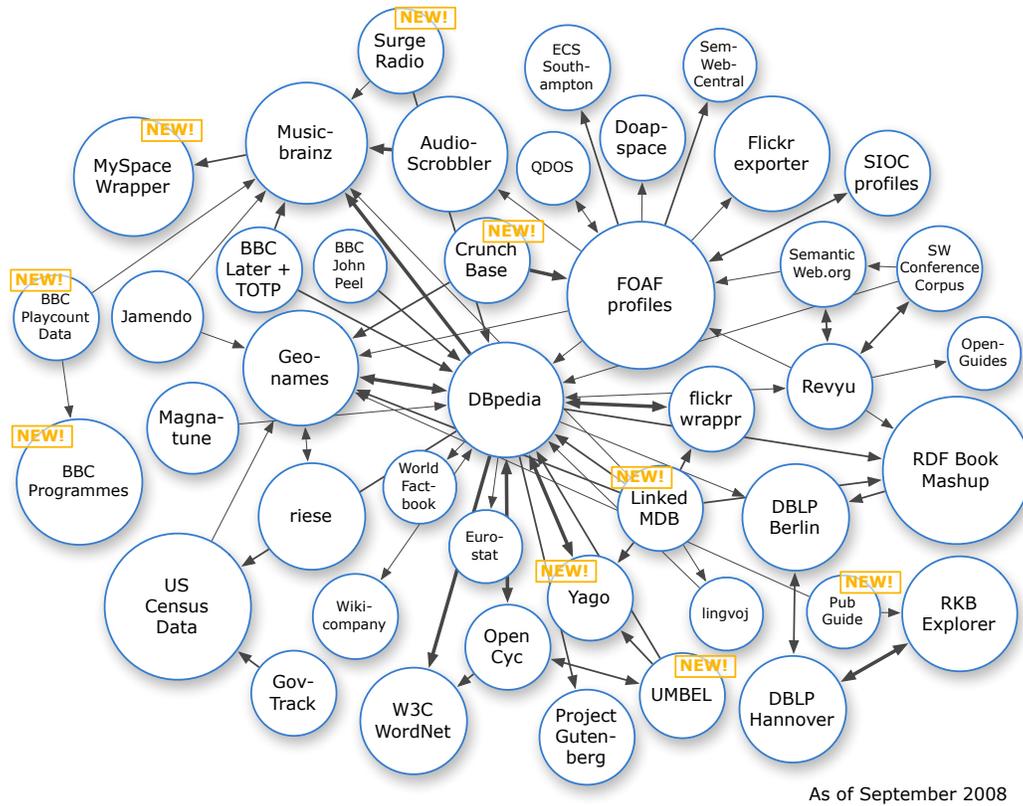


FIGURE 1.8 – Source image : Richard Cyganiak en 2008





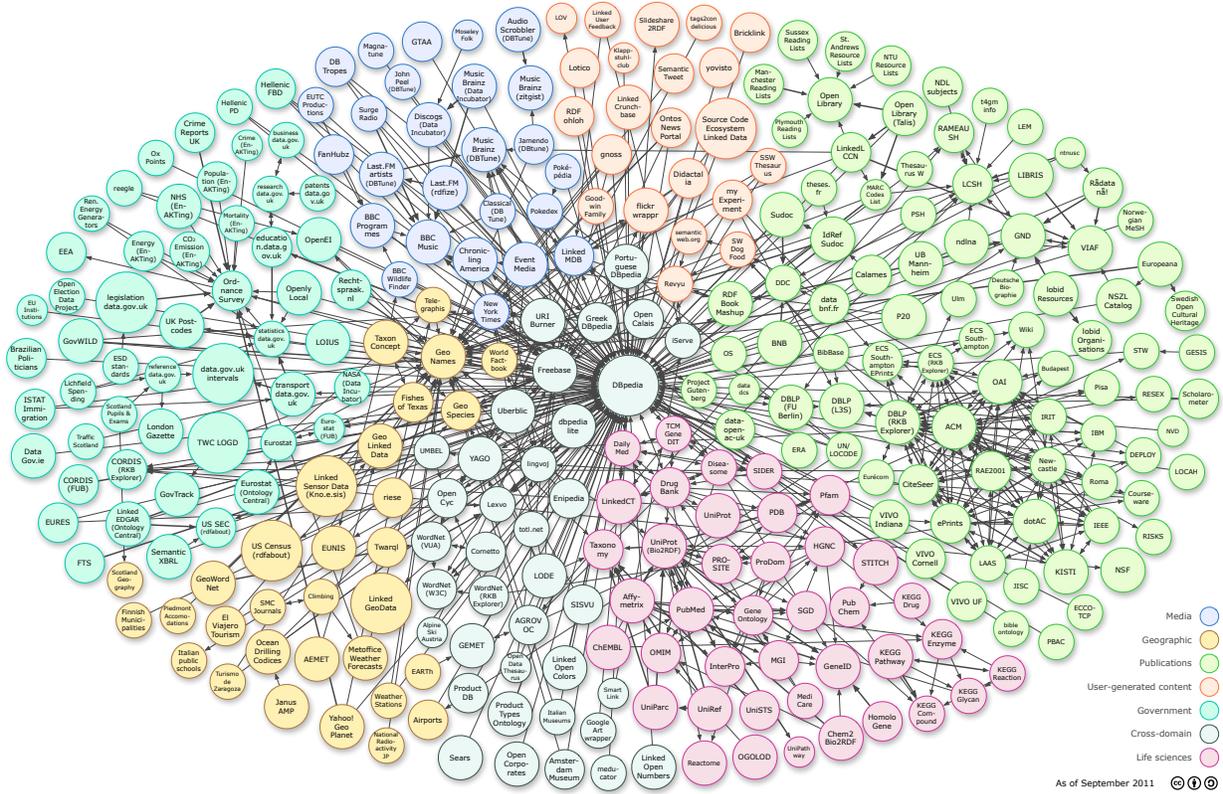


FIGURE 1.11 – Source image : Richard Cyganiak in 2011

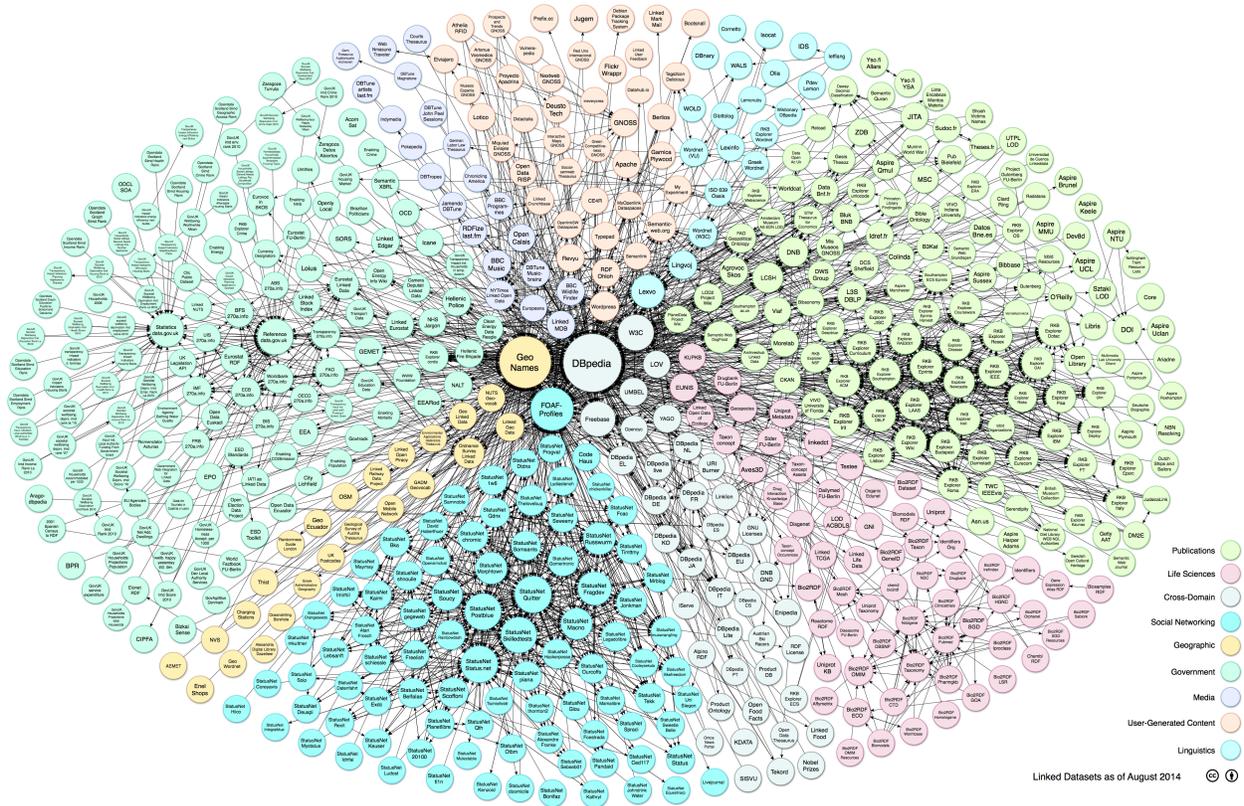


FIGURE 1.12 – Source image : Richard Cyganiak en 2014

## Rechercher et exploitation des données

- Annuaire des sources de données :
  - <http://thedatahub.org/>
- Moteur de recherche :
  - <http://sindice.com/>
- Navigateurs de données :
  - <http://graphite.ecs.soton.ac.uk/browser/> (navigateur simple)
  - <http://sig.ma/> (navigateur multi-source)
  - <http://www.visualdataweb.org/relfinder.php>

### 1.2.4 Divergences et convergences

- Microformats (Community)
  - <http://microformats.org/>
- The Open Graph protocol (Facebook)
  - <http://ogp.me/>
- Schema.org (Bing, Google, Yahoo)
  - <http://schema.org/>
  - <http://schema.rdfs.org/>

Ces projets ont le même objectif : rendre le Web plus accessible aux machines.

Bien qu'utilisant initialement des technologies différentes, ils convergent progressivement vers RDF.

## 1.3 Forger des IRIs

### 1.3.1 Problématique

Lorsqu'on publie des documents ou des données sur le Web, on est amené à « baptiser » la ressource correspondante, en lui *forgeant* un IRI.

---

**Note :** Le terme *forger* (en anglais *mint*) est une référence à la monnaie : un IRI est comme une pièce de monnaie, une marque (*token*) qui « vaut pour » autre chose.

---

### 1.3.2 Qui peut forger un IRI ?

N'importe qui ayant l'*autorité* sur un espace de noms :

- nom de domaine (e.g. `champin.net`),
- sous-espace (e.g. `http://liris.cnrs.fr/~pchampin/`)

La problématique du *nommage* des entités représentées se pose quel que soit le formalisme.

#### Interlude

'I don't know what you mean by "glory,"' Alice said.

Humpty Dumpty smiled contemptuously. 'Of course you don't – till I tell you. I meant "there's a nice knock-down argument for you !"'

'But "glory" doesn't mean "a nice knock-down argument,"' Alice objected.

'When I use a word,' Humpty Dumpty said in rather a scornful tone, 'it means just what I choose it to mean – neither more nor less.'

'The question is,' said Alice, 'whether you can make words mean so many different things.'

'The question is,' said Humpty Dumpty, 'which is to be master – that's all.'

– Lewis Carroll

### 1.3.3 Qu'identifie un IRI ?

Le triangle sémiotique

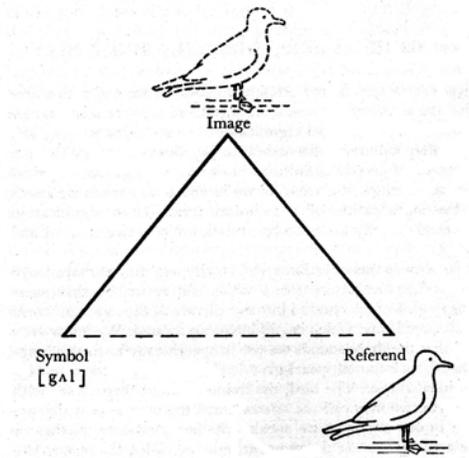


FIGURE 1.13 – Source image : Fred Andersson, d'après Simeon Potter

On peut assimiler :

- l'IRI au symbole
- la représentation à l'image
- la ressource au référent

Cf. *le troisième principe du Linked Data* (page 5)

---

**Note :** Bien sûr, on peut se poser la question d'identifier la *représentation* elle-même. Il existe des solutions.

---

### 1.3.4 Qui peut nommer une ressource

Les IRIs ont des propriétaires, mais pas (toutes) les ressources ; pour une ressource donnée (*e.g.* la Tour Eiffel), n'importe qui peut donc lui forger un IRI.

→ risque de prolifération de synonymes

Il existe une manière standard de déclarer la synonymie de deux IRIs (`owl:sameAs`, cf. *Vocabulaires et méta-vocabulaires* (page 33))

## 2.1 Introduction

### 2.1.1 Vue d'ensemble

RDF définit :

- une syntaxe abstraite (modèle de donnée),
- une sémantique pour *interpréter* la syntaxe abstraite,
- plusieurs syntaxes concrètes pour représenter/échanger la syntaxe abstraite.

## 2.2 Syntaxe abstraite et sémantique

### 2.2.1 Triplet

Toute information en RDF est représentée par un *triplet*,  
signifiant qu'une *chose* est en *relation* avec une autre.

Exemple :

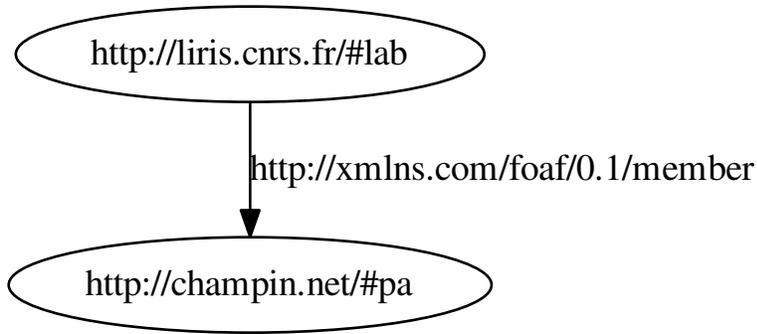
Le laboratoire LIRIS (sujet)  
a pour membre (prédicat)  
Pierre-Antoine Champin (objet)

### Nommage

Les choses sont nommées par des IRIs :

<http://liris.cnrs.fr/#lab>  
<http://xmlns.com/foaf/0.1/member>  
<http://champin.net/#pa>

On peut représenter ceci graphiquement :



### Remarque

Dans RDF, les URIs ne sont utilisés que comme des **identifiants opaques**.

Les représentations qui sont éventuellement accessibles via ces URIs (par *déréférencement*) n'ont aucune influence sur leur sémantique.

Notons que le mouvement *Linked Data* (page 5) préconise une utilisation *particulière* des URIs dans laquelle les représentations doivent être cohérentes avec la sémantique. Cependant cette contrainte n'est pas inhérente à RDF.

## 2.2.2 Préfixes

Pour simplifier les **notations**, on définit des préfixes courts correspondant à des préfixes d'IRI :

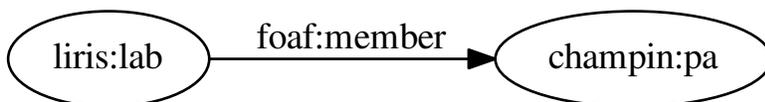
```

liris : → http://liris.cnrs.fr/#
foaf : → http://xmlns.com/foaf/0.1/
champin : → http://champin.net/#
  
```

On utilise ensuite des *noms préfixés* :

```
liris:lab foaf:member champin:pa
```

et également sous forme graphique :

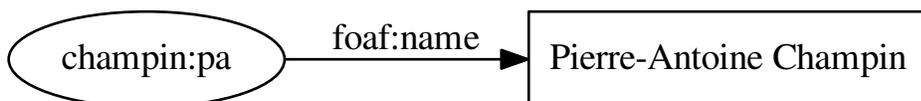


## 2.2.3 Littéraux

On peut également lier une ressource à une *donnée typée* (chaîne de caractère, entier, réel...), nommée un littéral.

```
champin:pa foaf:name "Pierre-Antoine Champin"
```

Traditionnellement, on représente les littéraux par des nœuds rectangulaires :



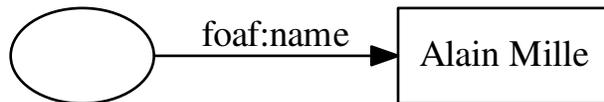
## 2.2.4 Nœuds muets

Enfin, RDF permet de parler d'une ressource sans connaître son IRI. Cela revient en logique à utiliser une variable quantifiée existentiellement.

(quelque chose) foaf:name "Alain Mille"

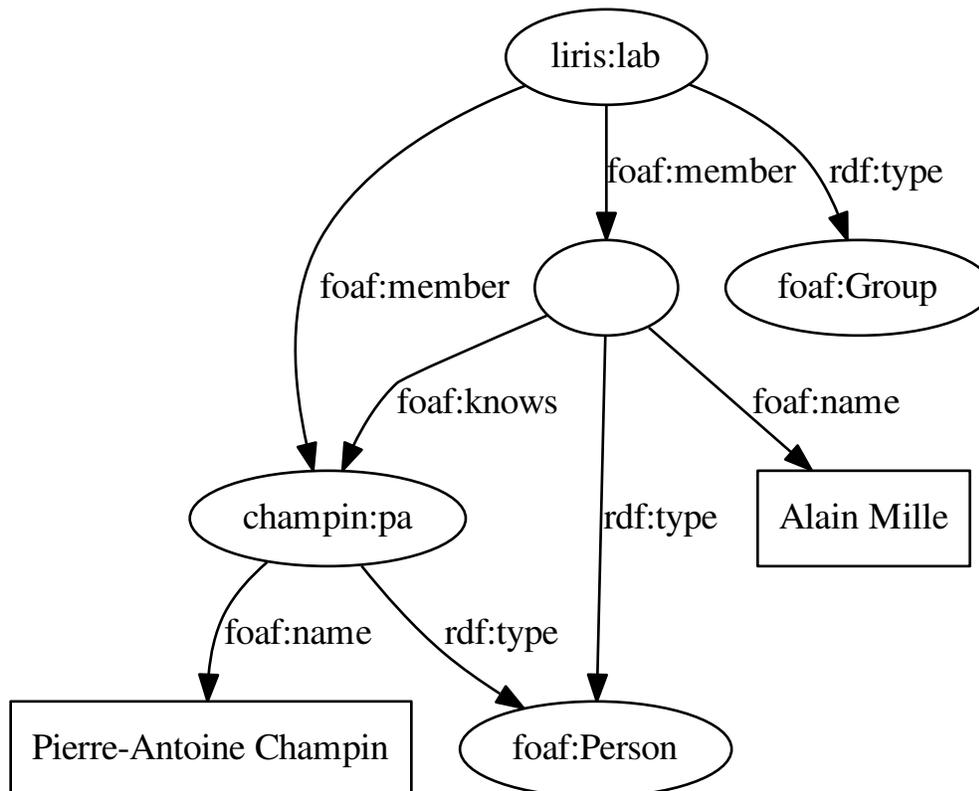
On parle alors de nœud *muet* (par analogie aux variables muettes).

Graphiquement, on représente cette ressource par un nœud vierge (*blank node*).



## 2.2.5 Exemple de graphe

Un ensemble de triplets forme un graphe orienté étiqueté.



## 2.2.6 Sémantique et inférences

En informatique, la *sémantique* d'un langage est souvent définie par les *inférences* (déductions) qu'on peut faire avec ce langage.

La structure du graphe permet de faire un minimum d'inférence, sans même avoir besoin de connaître le vocabulaire.

**Indice :** Par analogie, de l'arbre XML `<a b="foo" c="bar"/>`, on peut « inférer » l'arbre

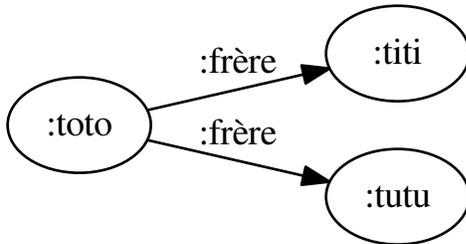
`<a c="bar" b="foo"/>` sans même connaître la sémantique des balises ou des attributs, car l'ordre des attributs n'est pas significatif.

---

### Monotonie

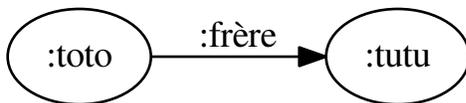
On peut déduire d'un graphe n'importe quel sous-graphe.

Si :



« toto a pour frère titi et tutu »

alors :

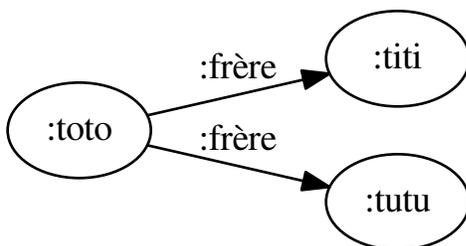


« toto a pour frère tutu »

### Anonymisation

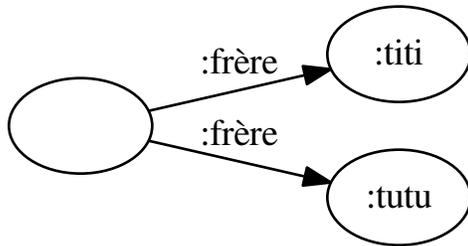
On peut remplacer n'importe quel nœud par un nœud muet.

Si :



« toto a pour frère titi et tutu »

alors :

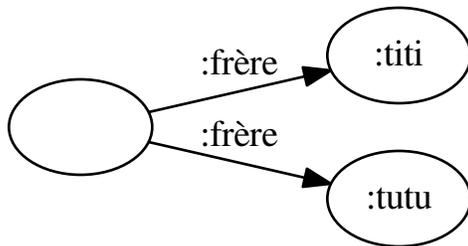


« quelque chose a pour frère titi et tutu »

### Éclatement

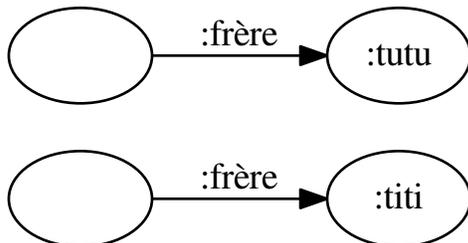
On peut « éclater » n'importe quel nœud muet.

Si :



« quelque chose a pour frère titi et tutu »

alors :

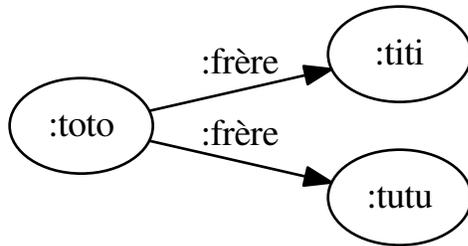


« quelque chose a pour frère titi, et  
quelque chose a pour frère tutu »

### Inférences supplémentaires

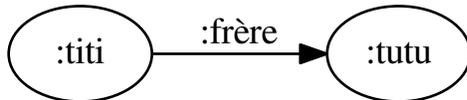
Bien sûr, des inférences supplémentaires peuvent être faites en prêtant une sémantique particulière aux IRIs utilisés dans le graphe.

Par exemple, si :



« toto a pour frère titi et tutu »

et si *frère* est une relation symétrique et transitive, alors (entre autres) :



« titi a pour frère tutu »

### Vocabulaire et sémantique additionnelle

On verra plus tard des langages (RDF-Schema, OWL) permettant de définir la sémantique de certains IRIs.

Mais ces langages ne peuvent pas remettre en cause les inférences présentées ci-avant.

---

**Indice :** Analogie : lorsqu'on définit un format XML, on prête une sémantique particulière aux éléments et attributs de ce format, mais on ne peut *pas* prêter de sémantique à l'ordre des attributs ;

- sémantiquement, ce ne serait plus du XML,
  - pragmatiquement, les outils standards (analyseur syntaxique, sérialiseurs) ne permettraient pas de contrôler cet aspect de la syntaxe.
- 

## 2.3 Syntaxes concrètes

---

**Note :** Dans cette section, on illustre chacune des syntaxes concrètes avec l'exemple de graphe donné *plus haut* (page 17).

---

### 2.3.1 RDF/XML

- syntaxe originale recommandée par le W3C (1999)
- basée sur XML
- relativement complexe et verbeuse

**Syntaxe** <http://www.w3.org/TR/rdf-syntax-grammar/>

**Valideur** <http://www.w3.org/RDF/Validator/>

#### RDF/XML : exemple

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        >
  <foaf:Group rdf:about="http://liris.cnrs.fr/#lab">
    <foaf:member>
      <foaf:Person>
        <foaf:name>Alain Mille</foaf:name>
        <foaf:knows
          rdf:resource="http://champin.net/#pa"/>
      </foaf:Person>
    </foaf:member>
    <foaf:member>
      <foaf:Person rdf:about="http://champin.net/#pa">
        <foaf:name>Pierre-Antoine Champin</foaf:name>
      </foaf:Person>
    </foaf:member>
  </foaf:Group>
</rdf:RDF>

```

### 2.3.2 Turtle : Terse RDF Triple Language

- dérivée du langage N3
- adoptée dans RDF 1.1 en 2014
- vise la simplicité et la compacité

**Syntaxe** <http://www.w3.org/TR/turtle/>

**Valideur** <http://www.rdfabout.com/demo/validator/>

#### Turtle : exemple

```

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix champin: <http://champin.net/#> .

liris:lab
  a foaf:Group ;
  foaf:member champin:pa, _:am .
champin:pa
  a foaf:Person ;
  foaf:name "Pierre-Antoine Champin" .
_:am
  a foaf:Person ;
  foaf:name "Alain Mille" ;
  foaf:knows champin:pa .

```

#### Turtle : exemple 2

```

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix champin: <http://champin.net/#> .

liris:lab
  a foaf:Group ;
  foaf:member champin:pa, [
    a foaf:Person ;
    foaf:name "Alain Mille" ;

```

```
    foaf:knows champin:pa .
  ] .
champin:pa
  a foaf:Person ;
  foaf:name "Pierre-Antoine Champin" .
```

### 2.3.3 RDFa : RDF in attributes

RDFa est une utilisation d'attributs (existants ou supplémentaires) de (X)HTML pour y inclure du RDF (à la manière des micro-formats) :

- facilite la migration de contenus HTML vers RDF
- facilite la maintenance en cohérence de la version HTML et des données RDF (*DRY : Don't Repeat Yourself*)

**Syntaxe** <http://www.w3.org/TR/rdfa-primer/>

**Valideur** <http://check.rdfa.info/>

**Distiller** <http://www.w3.org/2012/pyRdfa/>

#### RDFa : exemple

```
<p prefix="foaf http://xmlns.com/foaf/0.1/"
  about="_:am">
  <span property="foaf:name"
    >Alain Mille</span>
  est membre du
  <a rev="foaf:member" href="http://liris.cnrs.fr/#lab"
    >LIRIS</a> .
  Il connaît
  <span rel="foaf:knows" href="http://champin.net/#pa">
    <span property="foaf:name"
      >Pierre-Antoine Champin</span> ,
  un autre membre de
  <span rev="foaf:member" href="http://liris.cnrs.fr/#lab">
    ce laboratoire.</span>
</span> </p>
```

### 2.3.4 JSON-LD

- Rappel : JSON est un langage d'échange de données, basé sur Javascript, et très utilisé en développement web.
- JSON-LD (JSON Linked Data) permet d'interpréter une structure JSON comme du RDF,
- grâce à un *contexte* (implicite ou explicite).
- Objectif : faciliter l'adoption de RDF (syntaxe abstraite) auprès des développeurs d'applications web.

**Syntaxe** <http://www.w3.org/TR/json-ld-syntax/>

**Valideur** <http://json-ld.org/playground/>

#### JSON-LD : exemple

```
{ "@context" : { /* ... */ },
  "@id": "http://liris.cnrs.fr/#lab",
  "@type": "Group",
```

```
"member": [  
  {  
    "@id": "http://champion.net/#pa",  
    "@type": "Person",  
    "name": "Pierre-Antoine Champin"  
  },  
  {  
    "@type": "Person",  
    "name": "Alain Mille",  
    "knows": "http://champion.net/#pa"  
  }  
]
```

### 2.3.5 Autres syntaxes

- Comme l'illustrent RDFa et JSON-LD, tout langage peut être interprété comme du RDF :
  - dialectes en XML (GRDDL)
  - microformats (<http://http://microformats.org/>)
  - microdata (<http://www.data-vocabulary.org/>)
- ← Prépondérance de la syntaxe abstraite.
- Difficulté : faire correspondre des IRIs là où d'autres langages utilisent des termes « locaux ».



## 3.1 Introduction

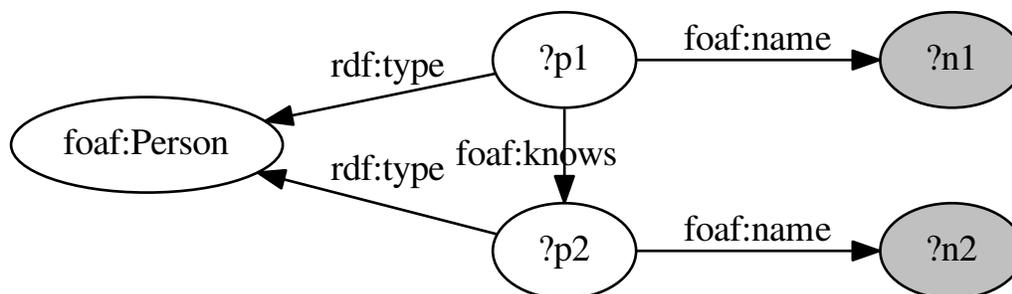
### 3.1.1 Objectifs

- Vous donner des bases pour écrire des requêtes SPARQL.
- Bonus : lire/écrire du Turtle (très proche de SPARQL).
- Ce n'est qu'une introduction ; pour en savoir plus :  
<http://www.w3.org/TR/sparql11-overview/>

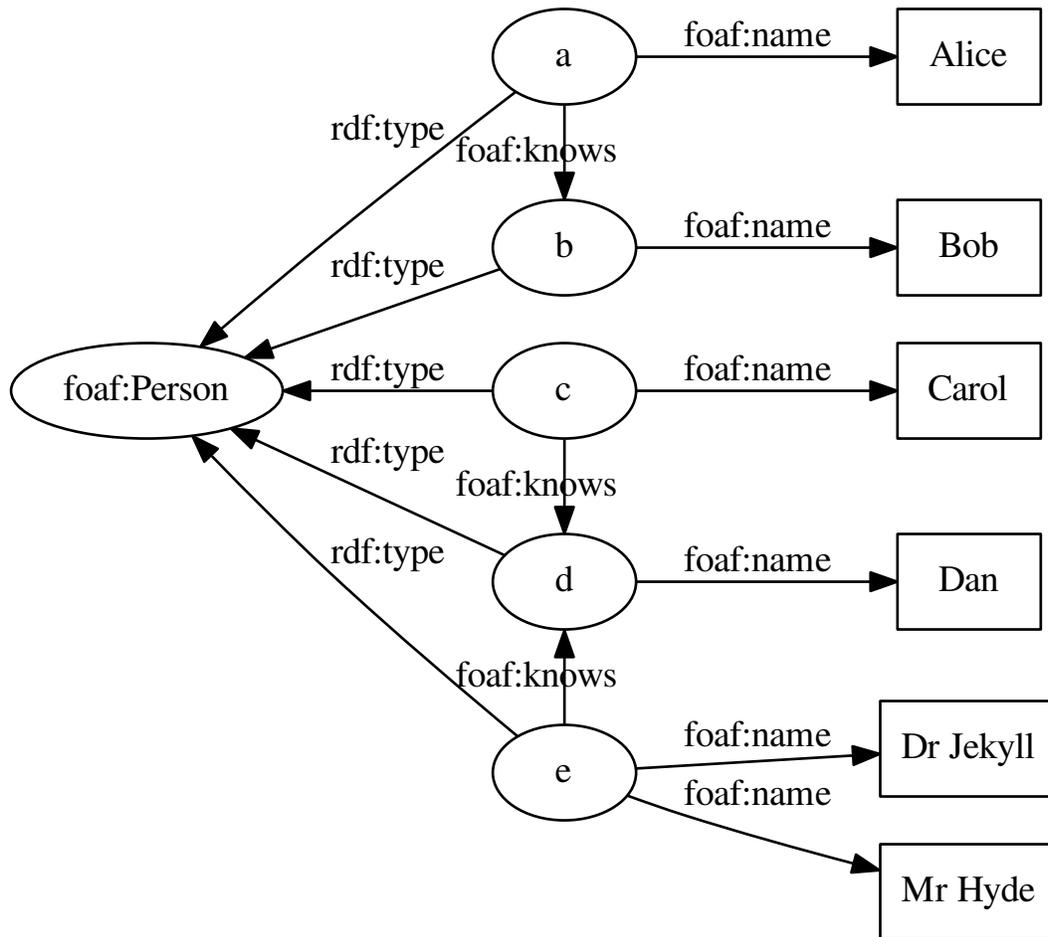
### 3.1.2 Requête simple

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?n1 ?n2
WHERE {
  ?p1 a <http://xmlns.com/foaf/0.1/Person> .
      foaf:name ?n1 ;
      foaf:knows ?p2 .
  ?p2 a foaf:Person ;
      foaf:name ?n2 .
}
```



Exemple de données



Résultats

n1	n2
Alice	Bob
Carol	Dan
Dr Jekyll	Dan
Mr Hide	Dan

## 3.2 Description du graphe requête

### 3.2.1 Préfixes

Rappel : les préfixes servent à abrégé les IRIs.

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX : <http://example.com/>
    
```

### 3.2.2 Termes

IRI en extension (relatif ou absolu) :

```
<http://xmlns.org/foaf/0.1/Person>
<../other-file.rdf>
<#something>
<>
```

IRI abrégé :

```
foaf:Person
:something
```

#### Termes (suite)

Litéral :

```
"Bonjour"
>Hello"@en          # avec tag de langue
"123"^^xsd:integer  # typé

42                  # equiv. "42"^^xsd:integer
1.5                 # equiv. "1.5"^^xsd:decimal
314e-2              # equiv. "314e-2"^^xsd:double
true                # equiv. "true"^^xsd:boolean
```

Nœud muet :

```
_:toto
[]          # voir ci-après
```

#### Termes (suite)

Variable (SPARQL seulement) :

```
?x
$y
```

NB : pas de distinction entre ? et \$, donc ?x et \$x identifient la *même* variable.

### 3.2.3 Triplets

— 3 termes (sujet, prédicat, objet) séparés par des espaces et suivis d'un point "." :

```
?p1 foaf:name "Pierre-Antoine Champin" .
```

— cas particulier : le mot clé "a" en position de prédicat est un raccourci pour  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> :

```
?p1 a foaf:Person .
```

— le retour à la ligne vaut pour une espace ; la structure est donnée par la ponctuation.

## Factorisation

- On *peut* « factoriser » plusieurs triplets ayant le même sujet en séparant les couples <prédicat, objet> par un point-virgule ";" :

```
<#pa> a foaf:Person ;
      foaf:givenName "Pierre-Anntoine" ;
      foaf:surname "Champin" .
```

- On *peut* « factoriser » plusieurs triplets ayant le même sujet et le même prédicat en séparant les objets par une virgule "," :

```
<#pa> foaf:phone <tel:+33-472-44-82-40>, <tel:+33-472-69-21-73> .
```

- On peut bien sûr combiner les deux types de factorisation.
- On n'est jamais obligé de factoriser, on peut aussi répéter les termes.

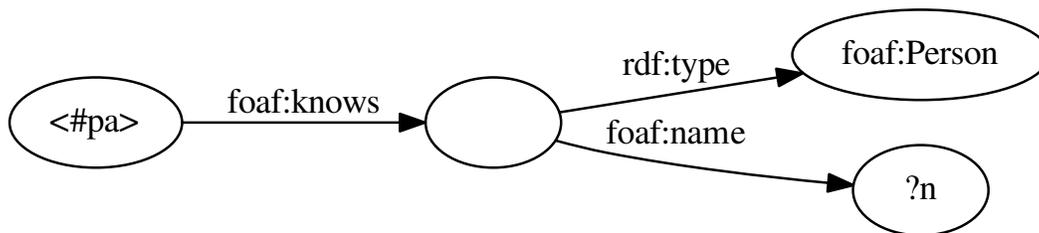
## Nœud muet

Lorsqu'un nœud muet n'a qu'un seul arc entrant, au lieu de lui inventer un identifiant local :

```
<#pa> foaf:know _:quelqun .
_:quelqun a foaf:Person ; foaf:name ?n .
```

on peut utiliser la notation [] :

```
<#pa> foaf:knows [
  a foaf:Person ;
  foaf:name ?n
] .
```



### 3.2.4 Sous-graphe optionnel

En SPARQL, on peut accepter qu'une partie du graphe ne soit pas satisfaite :

```
?p1 a foaf:Person ; foaf:name ?n .
OPTIONAL { ?p1 foaf:phone ?tel }
```

### 3.2.5 Filtres

En SPARQL, on peut ajouter des contraintes sur les valeurs d'un graphe, avec la clause `FILTER`.

```
?p foaf:age ?a .
FILTER (?a >= 18)
```

On peut combiner des conditions avec les opérateurs logiques « et » (&&), « ou » (||) et « non » (!).

```
FILTER ( 20 <= ?a && ?a < 30 )
```

#### Opérations utiles pour les filtres

- comparaisons : =, !=, <, >, <=, >=
- opérateurs arithmétiques : +, -, \*, /
- nature d'un nœud : isIRI, isBLANK, isLITERAL, isNUMERIC
- vérifier qu'une variable (utilisée avec OPTIONAL) a bien une valeur : Bound
- recherche de texte : REGEX(<variable>, <texte>)

## 3.3 Requête SELECT

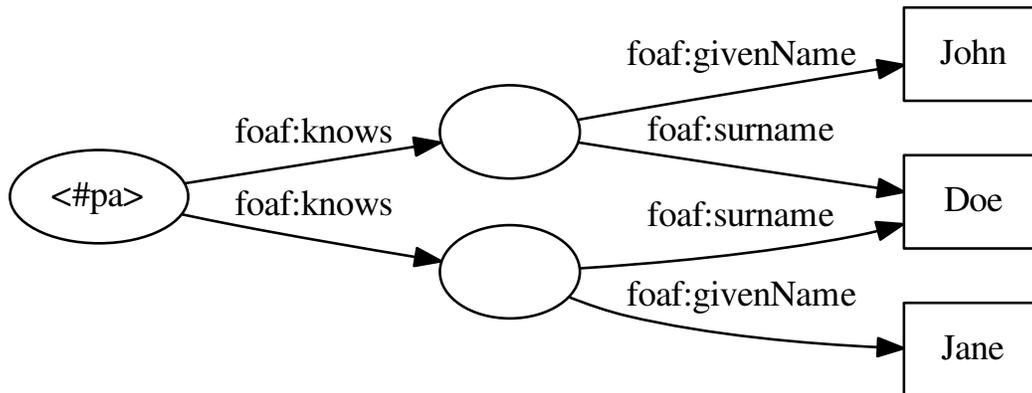
### 3.3.1 Présentation

- Similaire au SELECT de SQL :
  - projection sur un sous-ensemble des variables du graphe
- Résultat : tableau
  - une colonne par variable sélectionnée
  - une ligne par résultat
- Structure :
 

```
SELECT <variables> WHERE { <graphe> }
```

### 3.3.2 DISTINCT

```
SELECT DISTINCT ?sn
WHERE { <#pa> foaf:knows ?p. ?p foaf:surname ?sn. }
```



Sans le `DISTINCT`, la requête renverra deux fois le résultat `sn="Doe"`.

### 3.3.3 LIMIT et OFFSET

Pour obtenir les 10 premiers résultats :

```
SELECT ?p
WHERE { <#pa> foaf:knows ?p. }
LIMIT 10
```

Pour obtenir les 5 résultats suivants :

```
SELECT ?p
WHERE { <#pa> foaf:knows ?p. }
LIMIT 5 OFFSET 10
```

### 3.3.4 ORDER BY

```
SELECT ?p ?n
WHERE { <#pa> foaf:knows [ foaf:givenName ?p ; foaf:surname ?n ] }
ORDER BY ?n ?p
```

On peut aussi trier par ordre descendant :

```
SELECT ?name ?age
WHERE { <#pa> foaf:knows [ foaf:name ?name ; foaf:age ?age ] }
ORDER BY DESC(?age)
LIMIT 1
```

---

**Note :** l'utilisation du tri *et* de `LIMIT 1` permet ici de n'obtenir que la personne la plus vieille que connaît <#pa>.

---

### 3.3.5 GROUP BY

Sert à *aggréger* certaines valeurs avec l'une des fonctions d'aggrégations : Count, Sum, Avg, Min, Max, GroupConcat et Sample.

```
SELECT ?p1 (count(?p2) as ?cp2)
WHERE { ?p1 foaf:knows ?p2 }
GROUP BY ?p1
```

On peut combiner GROUP BY avec ORDER BY et LIMIT (attention à l'ordre) :

```
SELECT ?p1 (count(?p2) as ?cp2)
WHERE { ?p1 foaf:knows ?p2 }
GROUP BY ?p1
ORDER BY DESC(?cp2)
LIMIT 3
```

## 3.4 Autres types de requête

### 3.4.1 ASK

- Sert à demander si un graphe existe ou non dans la base.
- Résultat : vrai ou faux
- Structure :  
ASK { <graphe> }

### 3.4.2 CONSTRUCT

- Sert à construire un graphe à partir des résultat d'un autre
- Résultat : un graphe RDF
- Structure :  
CONSTRUCT { <graphe> } WHERE { <graphe> }
- Peut jouer un rôle similaire à XSL-T pour RDF

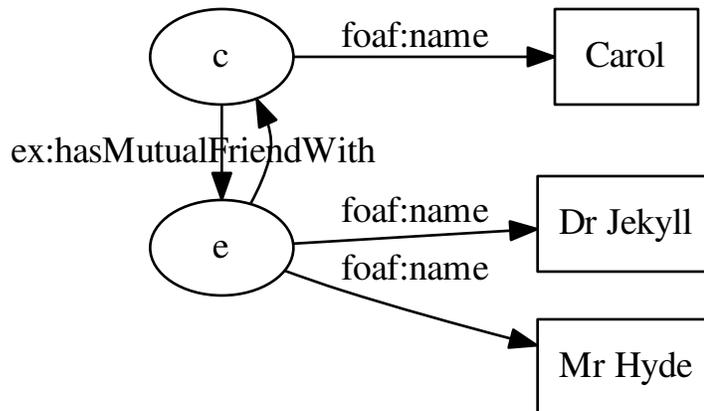
#### Exemple

```
CONSTRUCT {
  ?p1 ex:hasMutualFriendWith ?p3 ;
  foaf:name ?n1 .
  ?p3 foaf:name ?n3 .
} WHERE {
  ?p1 foaf:knows ?p2 .
  ?p3 foaf:knows ?p2 .
  OPTIONAL { ?p1 foaf:name ?n1 }
  OPTIONAL { ?p3 foaf:name ?n3 }
}
```

construit un graphe contenant des arcs `ex:hasMutualFriendWith` entre les personnes qui partagent au moins un ami, et copie également les arcs `foaf:name` concernant ces personnes, le cas échéant.

### Résultat de l'exemple

En appliquant l'exemple de CONSTRUCT ci-dessus au graphe donné en exemple en début de chapitre, on obtient :



### 3.4.3 SPARQL Update

Depuis la version 1.1, possibilité de *modifier* les données.

## 3.5 Quelques requêtes utiles

### 3.5.1 Exploration des types de ressources

```

SELECT ?type
WHERE { ?o a ?type }
GROUP BY ?type
ORDER BY DESC(count(?o))
LIMIT 30
  
```

### 3.5.2 Exploration des propriétés liées à un type

```

SELECT DISTINCT ?prop
WHERE { ?o a <http://example.org/UnType> ; ?prop ?val . }
LIMIT 30
  
```

---

## Vocabulaires et méta-vocabulaires

---

### 4.1 Vocabulaires

#### 4.1.1 Trouver un vocabulaire

- <http://lov.okfn.org/>
- <http://swoogle.umbc.edu/>

#### 4.1.2 Quelques vocabulaires utiles

##### Dublin Core

Méta-données à propos des documents :

- titre, résumé...
- créateur, contributeur...
- date de création, de dernière modification, versions...

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_dcterms.html](http://lov.okfn.org/dataset/lov/details/vocabulary_dcterms.html)

**homepage** <http://purl.org/dc/terms/>

##### FOAF : Friend of a friend

Description de personnes et de leur réseau social

- Personne (nom, prénom, page web, adresse e-mail, *connaissance*...)
- Groupe (membres...)
- Document (a pour sujet...), Image (représente...)

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_foaf.html](http://lov.okfn.org/dataset/lov/details/vocabulary_foaf.html)

**homepage** <http://www.foaf-project.org/>

##### SIOC : Semantically-Interlinked Online Communities

Description de communautés en ligne

- Forum, Blog, Wiki...
- Article, Commentaire...

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_sioc.html](http://lov.okfn.org/dataset/lov/details/vocabulary_sioc.html)

**homepage** <http://rdfs.org/sioc/spec/>

## WGS84 : Word Geodetic System

Coordonnées géographiques

- SpatialThing, Point
- latitude, longitude, altitude

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_geo.html](http://lov.okfn.org/dataset/lov/details/vocabulary_geo.html)

**homepage** (rdf) [http://www.w3.org/2003/01/geo/wgs84\\_pos](http://www.w3.org/2003/01/geo/wgs84_pos)

## GoodRelations

e-commerce

- Produit et Service, Offre...
- quantité, prix, garantie...

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_gr.html](http://lov.okfn.org/dataset/lov/details/vocabulary_gr.html)

**homepage** <http://purl.org/goodrelations/v1>

## Schema.org

Vocabulaire généraliste

- destiné à améliorer l'indexation des pages par les moteurs de recherche

**description** [http://lov.okfn.org/dataset/lov/details/vocabulary\\_schema.html](http://lov.okfn.org/dataset/lov/details/vocabulary_schema.html)

**homepage** <http://schema.org/>

## 4.2 Méta-vocabulaires

### 4.2.1 Motivation

Découverte de la sémantique d'un terme (IRI) en le déréréférençant.

Exemple : [http://dbpedia.org/resource/James\\_Bond](http://dbpedia.org/resource/James_Bond)

```
:James_Bond
  a dbo:FictionalCharacter ;
  dbo:creator :Ian_Fleming .
```

Ce principe s'applique également aux classes (e.g. `dbo:FictionalCharacter`) et aux prédicats (e.g. `dbo:creator`).

### 4.2.2 Problème

Syndrome du dictionnaire : il faut pouvoir s'« arrêter » sur des termes connus.

Nécessité d'un vocabulaire (ensemble de termes) permettant de décrire d'autres vocabulaires : **méta-vocabulaires**.

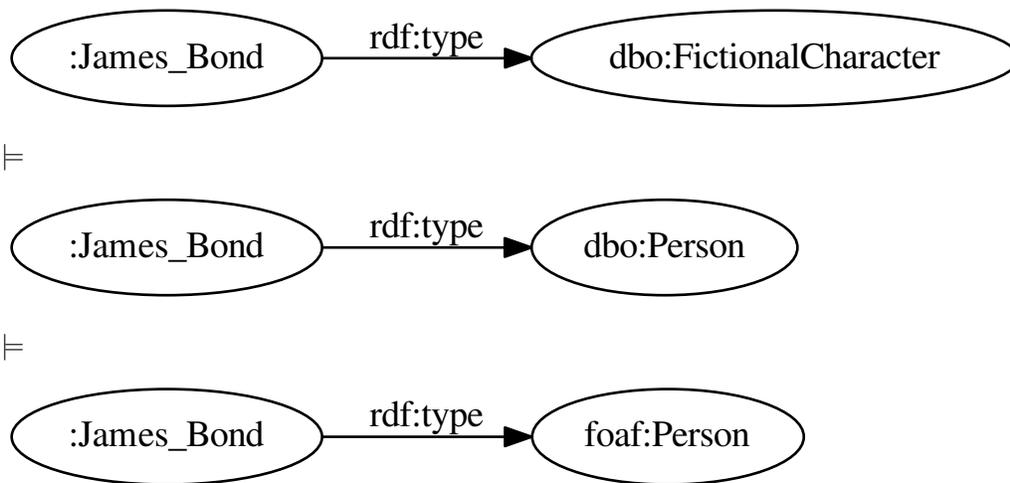
Analogie : XML-Schema est un vocabulaire XML dont la sémantique est connue *a priori*, et qui permet d'exprimer la structure de nouveaux vocabulaires.

### 4.2.3 Objectif

Expliciter *formellement* la sémantique des vocabulaires (en conformité avec la sémantique de RDF), afin de

- limiter les problèmes d’ambiguïté sur les termes
- permettre leur découverte dynamique
  - relations sémantiques internes
- assurer l’interopérabilité
  - relations sémantiques avec d’autres vocabulaires

### 4.2.4 Exemples d’inférences



## 4.3 RDF-Schema

### 4.3.1 Présentation

- RDF-Schema (ou RDF-S) est une recommandation du W3C publiée en même temps que RDF (1999 et révisée en 2004).
- Il permet d’exprimer une hiérarchie de classes et une hiérarchie de propriétés (relations).
  - hiérarchie au sens *large* : treillis
- Il permet aussi d’exprimer des contraintes sémantiques sur les propriétés et les classes.
  - /∧ contrainte sémantique ≠ contrainte d’intégrité

### 4.3.2 Espaces de noms et préfixes

Suite à des circonvolutions historiques, le vocabulaire RDF-Schema utilise deux espaces de nom, associés respectivement aux préfixes suivants :

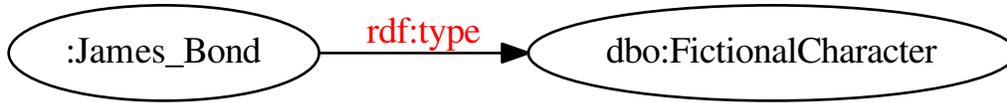
```

@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
  
```

### 4.3.3 rdf:type

`rdf:type` indique l'appartenance d'une ressource à une classe.

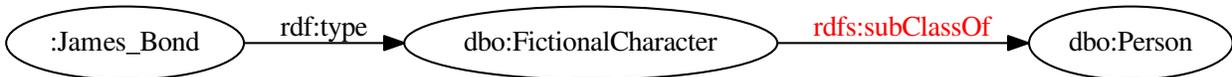
Exemple :



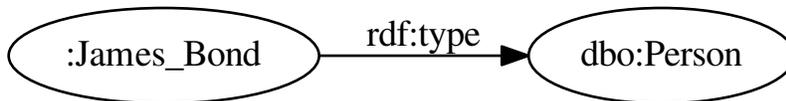
### 4.3.4 rdfs:subClassOf

`rdfs:subClassOf` indique une relation de spécialisation entre classes (« est une sorte de », ou « tous les X sont des Y »).

Exemple :



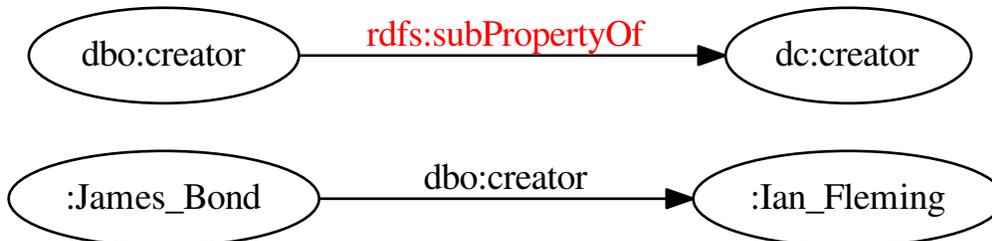
⊨



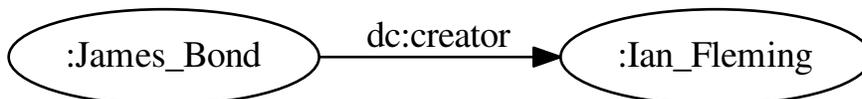
### 4.3.5 rdfs:subPropertyOf

`rdfs:subPropertyOf` indique une relation de spécialisation entre propriétés (« est une sorte de »).

Exemple :



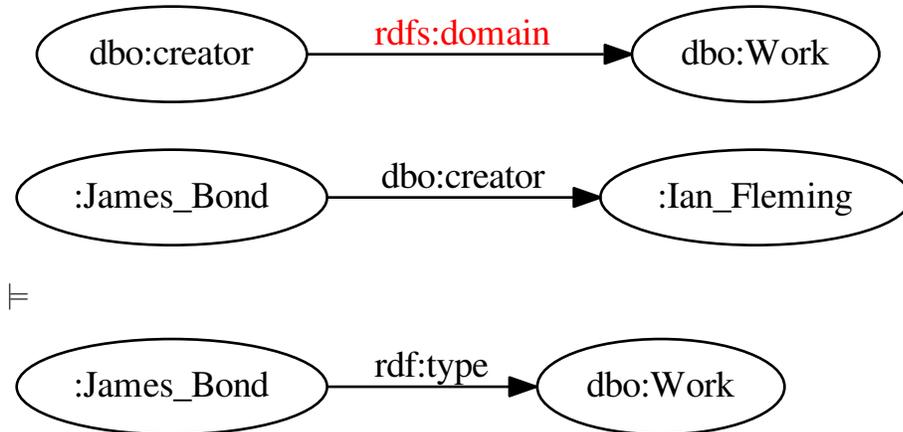
⊨



### 4.3.6 rdfs:domain

Indique qu'une propriété porte nécessairement sur les instances d'une classe.

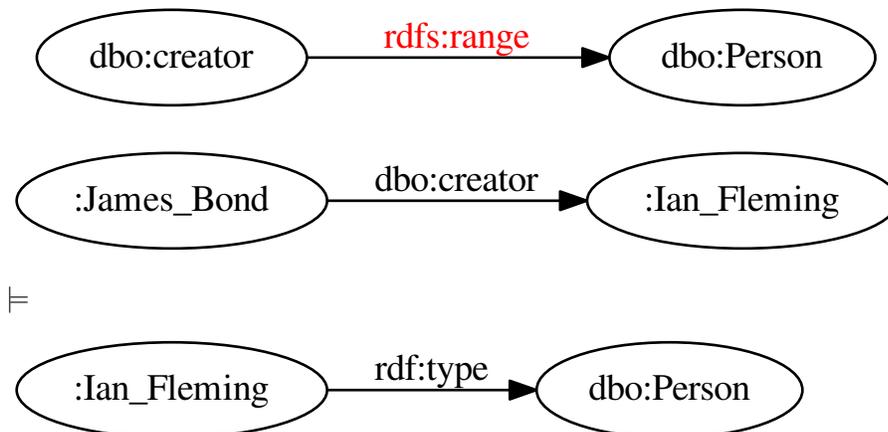
Exemple :



### 4.3.7 rdfs :range

Indique qu'une propriété a nécessairement pour valeur les instance d'une classe.

Exemple :



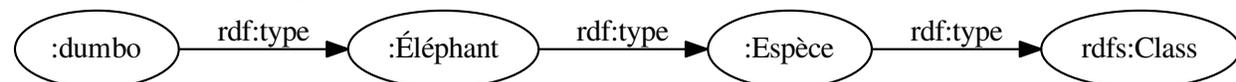
### 4.3.8 Documentation

RDF-Schema fournit aussi des termes pour *documenter* un vocabulaire :

- `rdfs:label` permet d'associer un libellé textuel à un URI (éventuellement plusieurs, par exemple dans plusieurs langues) ;
- `rdfs:comment` permet d'associer un commentaire textuel plus long ;
- `rdfs:seeAlso` permet de pointer vers une autre ressource.

### 4.3.9 Méta-modélisation

Rien n'empêche, en RDF-S, d'avoir une classe qui soit elle même une instance d'une autre classe (méta-classe). C'est d'ailleurs de cette manière que les classes sont identifiées.



### 4.3.10 Contrainte sémantique $\neq$ contraintes d'intégrité

Les méta-propriétés `rdfs:domain` et `rdfs:range` ne servent *pas* à vérifier qu'un graphe serait « valide ». Il ne permettent que d'*inférer* des faits supplémentaires.

- Comme RDF-S n'a pas de négation, ceci n'entraîne jamais d'incohérence *formelle*.  
 → en d'autres termes, la sémantique de RDF-S ne permet pas de détecter les incohérences (conceptuelles) que pourraient entraîner ces inférences.



## 4.4 OWL

### 4.4.1 Présentation

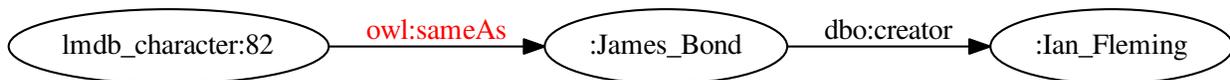
OWL (Web Ontology Language) a été recommandé par le W3C en 2004, et sa version 2 en 2009.

- C'est un méta-vocabulaire (comme RDF-S) inspiré des **logiques de descriptions** avec valeurs concrètes (littéraires).
- Il définit plusieurs *profils* offrant des compromis différents en terme d'expressivité et de complexité.
- Il mime les capacités de méta-modélisation de RDF-S (*punning*).

### 4.4.2 owl:sameAs

Indique que deux IRIs dénotent la même ressource.

Exemple :



⊨



---

**Projet 2014**

---

Votre travail consiste à proposer, par groupe de 2 à 4, un plan d'action pour la publication, sous forme de données liées en RDF, du jeu de données suivant :

Monuments d'Île-de-France (sources [data.iledefrance.fr](http://data.iledefrance.fr))

Ces données sont au format CSV, donc assimilables à une table dans une base de données relationnelles. Par plan d'action, on entend le fait de dégager les contraintes à respecter et les grandes lignes des solutions envisagées, pas le fait de réaliser en détail la publication.

- 1/ Vous décrirez d'abord le vocabulaire que vous proposez pour modéliser le domaine, sous forme d'un ensemble
- de classes (quelles sont les types d'entités qui existent ? — comme *CreativeWork*, *Book*, *Person*), et
  - de propriétés (quelles relations entretiennent ces types d'entités ? — comme *author* qui relie une entité de type *Person* à une entité de type *CreativeWork*).

Vous pouvez vous inspirer pour cela de la manière donc le vocabulaire Schema.org est présenté. Vous respecterez les bonnes pratiques de casse dans les dénominations des classes et des relations.

Vous ferez également des liens, quand cela est possible, entre les termes (URIs) de vocabulaire et ceux d'autres vocabulaires utilisés sur le web de données (au minimum Schema.org).

2/ Vous décrirez ensuite le principe de la conversion des données CSV (tabulaires) vers un modèle RDF (graphe) utilisant le vocabulaire décrit en 1/. Vous décrirez notamment comment forger des URIs pour toutes les entités décrites par le jeu de données (selon leur type). Vous donnerez un exemple du graphe généré à partir d'un petit extrait des données. Cet exemple mettra en jeu toutes les classes et les propriétés que vous aurez définies. NB : n'oubliez pas qu'il existe un outil en ligne permettant de dessiner le graphe correspondant à un tel exemple à partir d'une spécification en dot.

3/ Enfin, vous proposerez des méthodes pour lier (avec `owl:sameAs`) les différentes entités de votre graphe à des URIs de référence existant pour ces entités. Une source prometteuse de tels URIs de référence est la version française de *DBPedia* et la base nationale *Mérimée*. Vous identifierez notamment les rapprochements qui peuvent être (partiellement automatisés) et ceux qui seront faits manuellement.

Indication : pour trouver l'URI d'une entité sur *fr.dbpedia*, trouvez sa page sur Wikipedia France, et remplacer dans l'URI "fr.wikipedia.org/wiki" par "fr.dbpedia.org/resource".

4/ Individuellement, vous proposerez chacun 2 requêtes SPARQL (différentes de celles des autres membres de votre groupe) permettant d'interroger de manière pertinente ces données. Pour chaque requête, vous expliquerez en français la question à laquelle elle répond, et donnerez un exemple de résultat attendus. Ces requêtes peuvent éventuellement reposer sur des inférences, mais vous devrez alors le préciser.

Envoyez vos rapports (de préférence en PDF) à [pchampin@liris.cnrs.fr](mailto:pchampin@liris.cnrs.fr) pour le 12/01/2015.



## R

### RFC

RFC 1738, 2

RFC 2396, 2

RFC 3987, 2